情報システムにおける役割概念について(知識ベースとデータベースの統合化に関する研究)

AUTHOR(S):

小林, 要

CITATION:

小林, 要. 情報システムにおける役割概念について(知識ベースとデータベースの統合化に関する研究). 数理解析研究所講究録 1986, 593: 135-144

ISSUE DATE:

1986-06

URL:

http://hdl.handle.net/2433/99507

数理解析研究所講究録
593巻 1986年 135-144

情報システムにおける役割概念について

国際情報社会科学研究所　　小林　要

1．はじめに

情報システムは情報間の関係が定められたシステムであり、システムには目的があり構造と振る舞いがある。情報システムのクラスには、例えば、(a)人間社会における情報システムのクラス、(b)生物系における情報システムのクラス、(c)人間の知的な活動の側面における情報システムのクラス、(d)計算機システムにおける情報システムのクラス、(e)物理・化学・機械系における情報システムのクラス、などがあろう。

これらの情報システムを対象として、そのモデルを構築したり、あるいは設計したりする場合に、情報の構造化および情報システムの構造化が必要になる。これらの構造化の際には対象とする情報システムの“抽象化”（abstraction）が重要となる。抽象化の構造には様々なものがあるが、この研究では、「役割（role）」という抽象化の構造を分析する。

2．情報システムにおける抽象化の構造

抽象化の構造を取り上げた研究には様々なものがある。抽象化の構造を直接扱って、具体的な構造に役立ててきた分野の一つにソフトウェアの構造の研究がある。ここでは、ソフトウェアの構造に注目しながら、“抽象化の構造”について考察してみた。

まず代表的な例はWilkesが最初に提唱したといわれるsubroutineという概念であろう。subroutineによる抽象化の構造は、「計算や処理をroutine として一体化（aggregate）して主（main）と副（sub）の関係に構造化する」ことに相当する。

subroutineに見られる抽象化は、(1)処理をroutine とみなす抽象化と、(2)主に対する副の役割とみなす抽象化の２つの側面があったと解釈できる。前者は一連の手続きを一つに見なす「機能」の抽象化であったのに対し、後者は主と副という「役割」の抽象化であると言える。主と副という役割の抽象化は「上と下」の役割分担の構造に対応させられ、上下階層の階層構造へと構造化される。上下の関係は情報の粗と精との関係にも対応づけられ、抽象化の階層構造の在りかたを論じる出発点となる。

抽象化の階層（abstraction-hierarchy）を最初に論じたのはDijkstra〔1〕であると言われている。階層を意識することは、特定の階層がそれなりの纏まりを持つことを意味し、レベル（levels）自体が特定の意味階層を持つという抽象化がさらに入ることを意味する。抽象化レベル（levels of abstraction）という概念を用いる場合に抽象データ型（abstract data type）の概念が有効であることを主張したのはLiskov〔2〕である。

主と副という「上から見た」役割構造ではレベルを合わせるという工夫が必ずしも徹底

しない。そこで、ある特定レベルでの平坦な構造に着目して、その平坦なレベルでの役割構造を備える工夫が必要になる。Parnasは情報隠蔽（information-hiding）〔3〕などのモジュールの評価尺度を導入したのは、抽象化のレベルをはっきり分ける工夫の一つであったと考えられる。単なる抽象化という漠然としたものではなく、抽象化には情報を隠蔽する効果もあることを指摘したのである。

Parnas〔4〕はさらに、操作の抽象化にもクラスがあることを指摘し、o-functionやv-functionなどの「機能」のクラス分類を提唱した。これは「機能」の抽象化にも構造があることを示している。抽象データ型ではこれをoperation-cluster という、「操作の集団化」を図って、データ構造の抽象化とする試みを行っている。この場合、(1)インスタンスたちを生む母体としての「型」の抽象化と、(2)操作群の全体を規定する「機能」の抽象化と、(3)操作される対象の「構造」（データ構造）の抽象化、の少なくとも3つの抽象化が込められている。

さらに「役割」という側面で抽象データ型を見ると、“操作の全体”と“その個々の操作”という関連において、個々の操作の「役割」が付与される場合に、抽象的なデータ構造が浮かび上がる、という機構がある。これは必ずしも十分な指摘が無いが、データ構造としてのモデルを意識させるのに十分な「役割付与」がさらに必要であることに関係している。すなわち、個々の操作がどの「役割」を果たすのかが見える場合にはじめて効果的な抽象データ型になる。

型概念に近い抽象化の構造としてclass-instanceの抽象化の構造がある。これはプログラム言語SIMULA〔5〕に起源があるとされている。class において規定された諸機能がその各instacesに継承（inherit）されるという構造を持つ。今日のオブジェクト指向システムSmalltalk-80〔6〕等での中心的な抽象化の構造の一つである。

以上はプログラムの処理機能に焦点をあてた抽象化であるが、他方で、Hoare 〔7〕等による制御構造やデータ構造の「構造」の抽象化がある。すなわち、シーケンス、分岐、繰り返し、という3つの構造が基本である。いわゆる構造化プログラミング（structured-programming）で知られている。3つの構造で入出力データを特徴づけてプログラムを構成しようとする方法の一つに、Jackson Method〔8〕がある。HIPO〔9〕なども、モジュールを入力—処理—出力関係の構造とその階層構造として、構造の抽象化を図っている例であろう。しかし、このような「構造」の抽象化は「機能」の抽象化と異なり、入力—出力—上—下という構造的な役割認識が先行するため、機能の意味的なまとまりへの間接的な支援になっている。

制御の構造に着目した抽象化は並行処理系においても存在し、Ada のタスク概念〔10〕やHansenのモニタやプロセス〔11〕、SIMULAのco-routine〔4〕、ACTOR 理論〔12〕におけるactor やmessage-passing などがある。これらの内で、タスクやプロセス、コルーチンなどはサブルーチンに見られた主と副という役割分担の抽象化ではなく、兄弟や同僚と

しての役割分担の抽象化に近く、それらがどのように同期して動作すべきかが新たな課題となる。

Actor モデルにおけるactor は、メッセージを受け取ると動作が開始される構造を持っている。役割の抽象化という側面で捉えると、actor は行動の主体、あるいは俳優などの概念で捉えられ、役柄の付与ということに通じる側面を持つ点が興味深い。しかし、実際は機能の集合体としての抽象化をその内部に持ってしまう。オブジェクト指向システムにおいてもオブジェクトが行動の主体として機能の集合体をなす点で同様な性質を持つ。

抽象化の構造はデータベースモデル論の世界にも存在している。データモデルにおける抽象化は、プログラムにおいてみられるような処理機能の抽象化ではなく、プログラムで扱われる対象世界の抽象化に主眼がある。モデル化された情報システムは何らかのデータ構造にマッピングされ、データベース管理システムによる支援を受けることを前提とするため、機能や役割の抽象化の方法と構造の抽象化の構造との整合性が問題になり、初期のデータモデル論の多くは「構造」に重点が置かれた。

データモデル論に登場する初期の代表的な抽象化の構造にはCODASYL 委員会による情報代数（information algebra ）〔13〕に見られるような代数構造的抽象化とネットワークモデル〔14〕や階層構造モデル〔15〕に見られるようなアクセス構造的抽象化である。

関係データモデル〔16〕は構造の抽象化としてではなく、データの関係の抽象化に着眼した点が、抽象化の構造を議論するのに便利な道具となった。関係データモデルの構築における抽象化の構造としては、Chen〔17〕のEntity-Relationship-Model （実体—関連モデル）がある。これは、関係のクラスを実体関係と関連関係とに分ける方法を取り、データ間の依存関係を分離していく規範とするものである。この方法の中では「役割」概念を直接扱おうとする側面もある。

他方、CoddやFagin 等〔18,19 〕に始まるデータの正規形（Normal-Form ）の議論はデータ間の関数（function）的関係に着目してデータの分解と合成に関する議論に結びついた。これは、functionということばに代表されるように、データ間のいわば「機能的関係」の扱いであり、ここでいう「役割」に注目する概念とは別であろう。

関係データベースの設計方法に関連してSmith 等〔20〕は抽象化の構造に、一般化抽象（generalization abstraction）と一体化抽象（aggregation abstraction ）とがあることを示した。この指摘は人間の抽象化の構造にせまるものであり、抽象化の構造を議論する一つの重要な基盤を提供している。一般化抽象はClass-抽象の構造に近く、性質の継承がある抽象化の構造であるのに対し、一体化抽象は性質の継承は無く、一体となって初めてある対象を抽象する構造となっている。抽象データ型における抽象化の構造に対比させてみると、抽象データ型では型の性質が個々のインスタンスに継承される側面が一般化抽象に対比でき、抽象データ型のオペレーションクラスターによる抽象化の側面が一体化抽

象に対比できる。役割概念は後者の一体化抽象に関連しているものと思われる。

以上の考察から、ソフトウェアシステムにおける抽象化の構造の研究において、「機能」の抽象化と「構造」の抽象化と「役割」の抽象化の構造とがあったことが分かる。しかし、「役割」の抽象化の構造は必ずしも十分な検討がされてきたとは言い難い。

3．役割概念の構造

「役割」とはどのような抽象化の構造を持つであろうか。そもそも「役（role）」とはどのような意味であろうか。日本語で「役（やく）」という意味は、上役・下役などの「立場」とか、役場・役人などの「任務」や「しごと」、あるいは主役・脇役などの「劇における配役」の意味がある。役と書いて「えき」と読むと戦争を意味するが、これは読み方が違うので別にしよう。英語のroleにも、「劇における配役」の意味や「任務」の意味がある。

日本語の含蓄と英語の含蓄とを混同することはまずいが、両者に共通する面で捉えておくと、「何らかの場面における立場や役目を担っている」ことを意味していると解釈できる。そして、「劇における役」のような概念がその基礎にあるのではなかろうか。考えてみれば、劇という概念は古代からあるように思われる。

このような「役割」概念を特徴づけるには、「場面」とか「状況」などという概念が必要であることが分かる。場面や状況の裏には何らかの「問題設定」がある。場面や状況を構成する「対象と構造」も必要である。役という概念は、「そこに存在する実体に割り当てられる概念」であり、実体に内在する性質ではない。その場に応じて各々の実体の役割は異なる。例えば、部長という役割はその部の長としての立場を説明する概念であるが、その部長となっている実体としての特定の個人が生まれつき部長であったわけではない。部長という役割はその人に後から、その外から割り当てられた概念である。すなわち、役割と実体との関係はクラスとインスタンスという関係ではなくて、それぞれ独立な存在であり、たまたま「その実体（entity）に役割（role）が割り当て（assign）られていること」に本質がある。このことは、実体の世界（entity world）と役割世界（role world）とがそれぞれ独立に存在し、かつ、実体（entity）に対して役（role）が割り当て（assign）られている関係にある（図 1）。

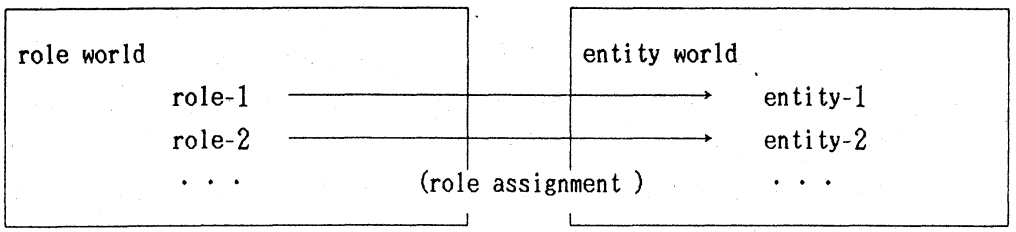

図 1：役割世界の実体世界への役割付与の構造

役割概念には問題設定が関与している。～したいという要請や、～という状態にするにはどうしたらよいかなどの問題設定がある。本来「システム」ということばに目的概念や評価尺度が付随していたことと合わせると、役割世界にはシステム概念が対応するとも考えられる。しかし役割概念と目的概念とは必ずしも一致しない。ある目的に対して、それを解決する手段としての役割概念もある。逆に手段概念が役割概念に相当するのではないか、との考察もあろうが、これも必ずしも一致しない。逆に、「目的」や「手段」や「方法」などの基本概念こそが役割世界における概念そのものであり、実体世界における実際の目的や手段や方法の実体に対して割り振られる役柄の「役目」を持っている。

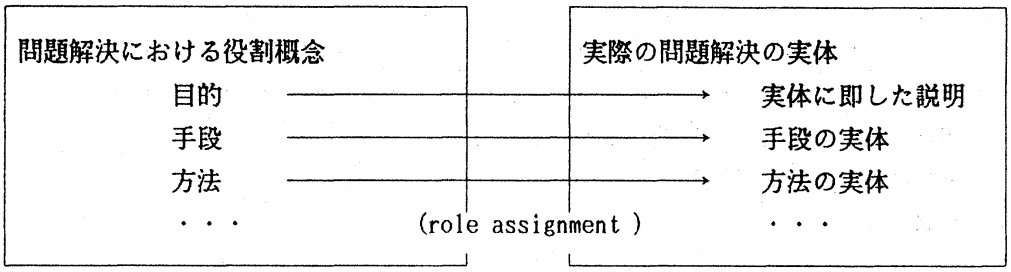

図 2：役割世界の実体世界への役割付与の構造の例 1

図 2のような場合に、役割世界は実体世界のクラス概念になっていて、実体世界はそのインスタンスではないのか、という疑問が起こる。しかし、インヘリタンス（性質継承）は実体において生じない。たとえば、よくあるカウンターを考える。X ：＝X+1 ；という処理をする場合、実体である変数X のクラスはカウンターであり、カウンターとしての性質としての機能特徴（入力・出力関係）は継承されるが、役割は変化してしまう。

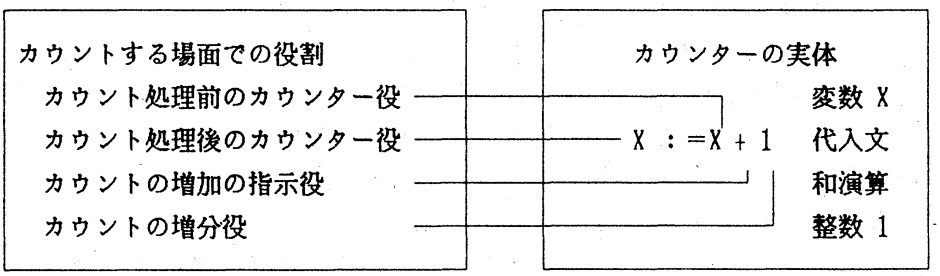

図 3：役割世界の実体世界への役割付与の構造の例 2

図 3の例で、役割概念には「登場する役柄の構成」という概念が必要であることが分かる。これは一種の一体化抽象（aggregation abstraction ）ではなかろうか。それも、カウントする、という目的や問題意識に対して合目的的に構成される役柄から構成される。すなわち、問題を解こうとする問題解決姿勢から生まれた役柄でもある。このことは、役柄の間における関係付けと、問題解決のための解法とが何らかの形で結びつくべきことを示している。役割図式はいわば「あるべき姿のモデル」として働く側面を持っている。図 3の例はそうした問題解決の図式を生むための側面であり、次の図 4のような全体図式の

一部であったと解釈される。

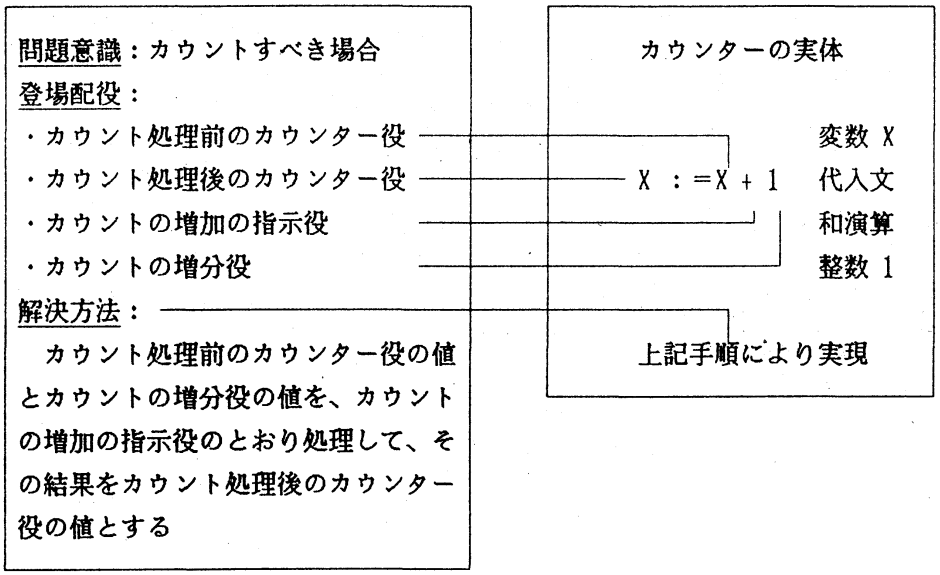

図 4：役割世界の実体世界への役割付与の構造の例 3

「こうあるべき」とする世界と「このようになっている」という世界との相違としての「役割世界」と「実体世界」の相違は、「理想」と「現実」の相違に近く、「実体世界」がいつも「役割世界」のようになるとは限らない。したがって、役割付与（role assignment）がいつもできるとは限らない。そこで、人間は同一の実体世界に対して、役割世界を取り替えながら問題を解いていくのではなかろうか。

役割付与が役割世界と実体世界との２つの世界の対応づけであると見ると、役割概念を考えることは類推〔21〕の問題のようにも思える。確かに，アナロジー概念を利用している側面があるが、役割概念では「似ているかどうか」よりも、「どうするべきか」を求める点が必要である。問題意識から役割世界を持ってきて、積極的に役割付与を試みて、図式が対応すれば解決の方策を立てていけることを要請しているのである。

状況に関係するという意味では、状況意味論〔22〕に深い関係があると考えられる。ただし、ここで考えようとしている役割概念は、状況意味論で定義されたroleと非常に類似しているが、必ずしも一致しないのではないかと考えられる。これは今後の研究によって明らかにされてくるものと考える。

内在する性質を内包に対応させるとすれば、外在する性質である役割概念は外延に関連することになる。しかし、どのように関連するのかは現在では明らかでない。今後に残された課題は多い。しかし、この「役割」という概念は我々に大きなヒントを与えているのではなかろうか。

4．役割抽象（role abstraction）

人間は役割世界をどのように構築しているのであろうか。クラス化の抽象やフレームへの抽象化〔23〕などとどのように違うであろうか。これにはもうすこし例題を考察する必要がある。ここではハノイの塔を例にとろう。ここではハノイの塔それ自身に興味はないし、一つの問題解決場面の例として取り上げるので、２枚の円板だけからなる場合を例にしよう。

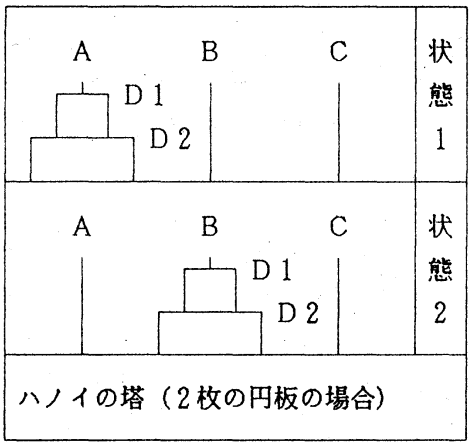

図 5：２枚円板のハノイの塔の問題

問題はよく知られているので、詳細は略すが、図 5に示すように、２枚の円板が棒Ａにあり、これらを棒Ｂに移動するが、一度に一枚の円板しか動かせず、かつ、小さい円板の上に大きな円板を乗せることはできない。円板はかならずどれかの棒に刺さないといけない。

登場する実体は、棒Ａ，棒Ｂ，棒Ｃ，円板Ｄ１，円板Ｄ２，それに地面である。ようするに状態１から状態２にする手順を求められているのである。

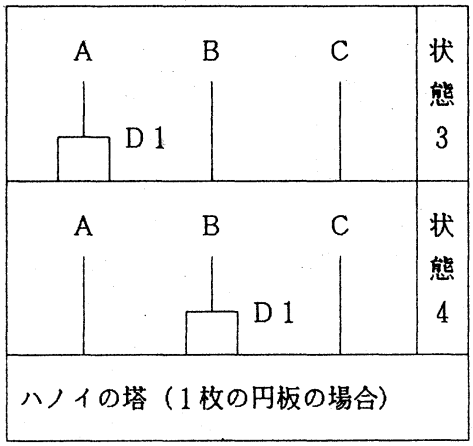

図 6：１枚円板のハノイの塔の問題

この問題を解くには、１枚の円板の移動問題が解けていなければならない。そこでまず１枚の円板の問題を図 6に描こう。

この場合の役割世界は、移動される円板と移動元の棒と移動先の棒という配役で決まる。もちろん移動される円板は移動元の一番上にある円板であり、移動先の棒には移動される円板よりも小さい円板は無いから、移動の条件も満足されている。したがって、その答えは「移動元の棒にある移動される円板を移動先の円板に移動する」である。

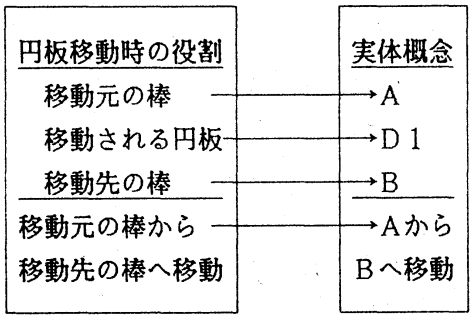

図 7：役割図式その１

この役割を実体に付与すると実際の答えである「Ａにある円板をＢに移動する」が対応して得られることになる。

ここで重要なことは、役割概念は問題の性質から導かれていることである。すなわち、「移動する」という目的に対して、「

移動される対象」、「移動元の場所」、「移動先の場所」、「移動元から移動できる条件」、「移動先へ移動できる条件」、などの検討項目が必要になることが導出されていなければならないのである。

さて、2枚円板の場合に戻ろう。2枚円板の実体と図 7の役割図式との対応を取ろうとすることから始める。すると、移動される円板の役割を果たしている実体が、D1とD2の2枚の円板であることが分かる。そのまま答えを求めると、「Aにある2枚の円板をBへ移動する」が答えである。ところが、一度に2枚を移動することはできない、という条件が働き、これは正解ではないことがわかる。このことは、移動される円板の役割をさらに分割しなければならないことを意味する。そこで、「上の円板」「下の円板」という役割に分割したとする。

ここで、さらに上の円板を主役として移動しようとすることを考えよう。すると、状態1では上の円板がすぐに棒Aから外せることは分かるが、移動先の棒には、下の円板がそれより先に刺されていなければならないことになる。このことは、上の円板よりも下の円板のほうが先に主役にならなければならないことを意味する。

そこで、下の円板を主役にする。すると、状態1では、上の円板がじゃまになる。上の円板はじゃま者の役割を果たしているのである。移動目的に対してじゃま者があれば、そのじゃま者をじゃまでない場所に退くことが必要になる。じゃま者は上の円板である。上の円板をじゃまでない場所に退くことが目的となる。ではどこがじゃまでない場所になるのか。それは移動元でもなく、移動先でもない、作業用の棒である。したがって、上の円板を作業用の棒に移動させてから、下の円板を移動先の棒に移動するとよい。

さて、上の円板を主役として見た時の要請がこれで満足されたので、それまでじゃま者であった上の円板がすなおに主役となる。棒Aから棒Cに移動しているので、移動元とする棒の実体は代わったが、移動元の棒から移動先の棒へ移動させるという一枚円板の役割図式がそのまま当てはまり、完了する。

移動問題における役割	実体
移動主役	D2
移動元	棒A
移動先	棒B
じゃま者	D1
じゃまでない場所	棒C
じゃま者をじゃまでない場所に移動してから移動	Aからc

図 8：役割図式その 2

ここで注目すべき点は、移動問題における役割図式が一様でない点である。実体である円板や棒は、あるときは移動元の棒になったり、あるときは移動先の棒になる。あるときは主役であり、あるときはじゃま者扱いされるのである。

役割は実体に付与されるが、実体それ自体が本来所有する性質ではなく、状況と目的とに照らして付与されるものである。

役割図式の形式はフレーム理論に類似している点も多い。しかし、フレーム理論では、役柄と実体とのペアの集合として規定された概念であるのに対して、ここでは実体は実体なりの一般化抽象されるべきものとして捉え、役割概念も独立に存在し、役割付与はインスタンシエーションではなく、積極的な対応づけの仮説になっている。すなわち、データ実体を説明するスキーマとしてのフレーム概念に対して、データ実体に対する解釈仮説としての役割図式なのである。よって、実体に合わなければ役割概念のほうが棄却される。フレームではフレームに合わないデータ実体が棄却されてしまう。役割概念という外枠のほうが仮説なのである。理想は仮説であり現実によって棄却されるという真理に基づく考え方をこの役割図式では用いている。

5．おわりに

情報システムにおける抽象化の構造として「役割抽象（role abstraction）」という構造のあることを示した。役割抽象には、問題、状況、役柄などの概念があり、問題解決の指針を与える役割がある。従来の抽象化の構造に比較すると、「あるべき」とする世界と「こうある」とする世界とを分離し、前者を役割世界、後者を実体世界とみなす点が異なり、特に、実体世界における問題を解決する場合の仮説として役割世界を用いると効果がある。「あるべき」とする仮説の役割とは、棄却される対象であり、実体世界を真実の世界とする立場から発想されたものである。

役割概念を用いることによって、情報システムの効果的な設計や保守作業ができるようにすることがこの研究の最終目標である。ソフトウェア開発の自動化に役立てるにはまだ不十分であるが、従来の抽象化技術だけではさらに不十分であろう。

参考文献

〔1〕Dijkstra,E.W. : The Structure of THE Multiprogramming System,CACM,VOL.11, No.5,pp.341-346,May,1968.

〔2〕Liskov,B.H. and Zilles,S.N. : Specification Techniques for Data Abstractions,IEEE Trans.on Software Engineering,Vol.SE-1,No.1,pp.7-19,March 1975.

〔3〕Parnas,D.L. : On the Criteria to be Used in Decomposing Systems into Modules,CACM.Vol.15,No.12,pp.1053-1058,December 1972.

〔4〕Parnas,D.L. : A Technique for Software Module Specification With Examples, CACM.Vol.15,No.5 ,pp.330-336,May 1972.

〔5〕Dahl,O.J.,Nygaard,K. : Simula - an Algol-Based Simulation Language,CACM, Vol.9,No.9,September 1966.

〔6〕Goldberg,A. and Robson,D. : Smalltalk-80,The Language and Its Implementation,Addison Wesley,1983.

〔7〕Dahl,O.J.,Dijkstra,E.W. and Hoare,C.A.R. : Structured Programming,Academic Press,London,1972.

〔8〕Jackson,M.A. : Principles of Program Design,Axademic Press,New York,1975.

〔9〕 IBM : HIPO : Design Aid and Documentation Tool ", IBM, SR20-9413-0, 1973.
〔10〕 US DoD : Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815 A, January 1983.
〔11〕 Hansen, P.B. : The Architecture of Concurrent Programs, Prentice-Hall, 1977.
〔12〕 Hewitt, C., et al. : A universal modular actor formalism for artificial intelligence, IJCAI 3, 235-245, 1973.
〔13〕 CODASYL : An Information Algebra Phase I Report-Language Structure Group of the CODASYL Development Committee, CACM, Vol.5, No.4, pp.190-204, 1962.
〔14〕 Bachman, C.W. : Data Structure Diagrams ", DATABASE (ACM SIGBDP Newsletter) Vol.1, No.2, pp.4-10, 1969.
〔15〕 IBM : IMS /VS Application Programming Reference, SH 20-9026
〔16〕 Codd, E.F. : A Relational Model of Data for Large Shared Data Banks, CACM, Vol.13, No.6, pp.377-387, 1970.
〔17〕 Chen, P.P. : The Entity Relationship Model - Toward a Unified View of Data, VLDB 75, also in ACM TODS, Vol.1, No.1, pp.9-36, 1976.
〔18〕 Codd, E.F. : Further Normalization of the Data Base Relational Model, in Courant Computer Science Symposium 6, Prentice-Hall, Data Base Systems, pp.33-64, 1972.
〔19〕 Fagin, R. : Multivalued Dependencies and a New Normal Form for Relational Database Design, VLDB 77, pp.441-446, 1977.
〔20〕 Smith, J.M., and Smith, D.C.P. : Database Abstractions : Aggregation and Generalization , ACM TODS Vol.2, No.2, pp.105-133, 1977.
〔21〕 Haraguchi, M. : Towards a Mathematical Theory of Analogy, Bull. Information and Cybernetics, Vol.21, pp.29-56, 1985.
〔22〕 Barwise, J., and Perry, J. : Situations and Attitudes, MIT Press, 1983.
〔23〕 Minsky, M. : a framework for representing knowledge, in The psychology of computer vision (Winston ed.) , McGraw-Hill, pp.211-277, 1975.